Alessio Ferraro / CS430 - Spring 2019 / Professor Bo Sheng

Spectrum Spinner Final Report

**Project Statement:** My project is a game titled Spectrum Spinner. In my game, the player must drag their finger across the screen to move a bar near the bottom of the screen with various colored segments in order to match the color of balls falling from the top of the screen. If the player successfully matches the color of the ball with the color of the bar segment, they will be given a point and the game continues. If the colors do not match, the game will end. The color of the ball is determined by an RNG from a set selection of colors. As the game continues, it will get harder and harder. Every 5 points the ball will get slightly faster and every 10 points the number of colors on the bar and that the ball can be will grow. At first, there will only be red and blue, but green, magenta, yellow, and cyan will be added (in that order) as the game goes on.

**Application Design:** The application consists of 5 components: MainActivity, GameplayActivity, GameplayView, GameoverActivity, and DBHelper.



*Activity Flow*: When the user first opens the app, they will start on MainActivity. When GameplayActivity is opened by the player, MainActivity will remain paused in the background. GameplayActivity’s layout xml merges GameplayActivity with GameplayView, so GameplayView will also be started and the two will run at the same time, both visible on the screen and layered atop one another. When the game ends, GameoverActivity will be started in the foreground. GameplayActivity will be pushed into the background then immediately destroyed. When the player destroys GameoverActivity, MainActivity will be resumed.

*MainActivity*: The main menu of the game; displays the title and current system high score. On resume (when the app is first opened or GameoverActivity is destroyed), DBHelper will be called to retrieve the high score from the system database. When the “Start” button is pressed, the system will create and intent holding the high score and launch GameplayActivity with that intent. When the “Reset” button is pressed, DBHelper will be called to set the high score in the system database to 0.

*GameplayActivity*: Displays all the text-based UI elements (score and high score), tracks the positions of the player’s finger on the screen and runs the thread and all the calculations needed for gameplay. The important variables of this activity are: the score, the high score, the position of the enemy ball, the position of the player bar, the color of the current ball (represented by a number: 0 for red, 1 for blue, 2 for green, 3 for magenta, 4 for yellow, and 5 for cyan), the color of the next ball, and the player’s rank (denoting how many colors can selected at this time). When first launched it sets the player’s score to 0, the rank is set to 2 (just red and blue), the high score is set to the number gotten from MainActivity’s intent and then the gameplay thread is launched. Every run, the thread:

Is the game still active? If yes:

- Enemy position + 2 + (score/5)

- Is enemy position >= 85? If yes:

- Is player finger in place for the ball to touch the right section of the bar?

-If no, then end game

- Score + 1

- Enemy position = 0

- Current color set to stored next color

- Next color determined by RNG (number between 0 and rank -1)

- Is the score a multiple of 10?

- If yes, and the score is below 6, rank + 1

- Sleep thread 100 milliseconds

- Update GameplayView

When the game is ended, stop the thread, create an intent with the game score and last high score, and launch GameoverActivity with the intent and then destroy GameplayActivity.

*GameplayView*: The canvas that all graphic elements of the UI are painted on. A ball in the top right corner of the screen is painted the color of the next color. The enemy ball is painted the current color along a line going down the middle of the screen on a position 0 - 85 “units”. A single unit represents 1% of the screen’s length, with 0 units being the top of the screen and 85 units being 85% of the way down the screen where the player bar is located.

0 Units

85 Units

Guide Ball

Guide Bar

The player bar is a series of colored rectangles drawn side by side. The bar itself is twice the length of the screen so that as you drag your finger across the screen it will appear that the sections dragged off the edge of the screen are coming out of the other side of the screen, creating the illusion that colors are rotating on a wheel. The player’s finger motion is translated to movement of the bar by having the middle red section follow the position of the player’s finger. The bar includes two extra pieces of UI to help the player: a black ball above the middle red section of the bar (to help players keep track of the section they control), and a screen-width white bar under the white bar centered at the player-controlled red section (as not the entire bar can reach the middle of the screen touch the ball, the white bar tells the player the difference between usable sections of the bar and those there for aesthetics only). As the player ranks up more colored sections are added to the bar as the older ones shrink.

**Rank 2**

**Rank 3**

**Rank 4**

**Rank 5**

**Rank 6**

*GameoverActivity*: The end-of-game screen; displays the score of the finished game and displays a message if the player got a new high score. If the player beat the old high score that game, then DBHelper will be called to set the higher score as the new high score in the system database. When the “Finish” button is pressed, GameoverActivity will be destroyed.

*DBHelper*: If there is no table for Spectrum Spinner in the system database, it will use SQLlite statements to create a 1-column table and put a new row with a 0 in the column. When asked to get the current high score, a SQLlite statement will be used to retrieve the value of the column. When asked to alter the high score, a SQLlite statement will be used to replace the value of the column to the given number.

**Application Evaluation:** I tested the application by loading the app into the emulator and playtesting it. If I needed to test anything involving the rank system I would go into the code and set the starting score or starting rank as needed. There are no known major issues to fix.

**References:** For this project I referred to heavily and at times ripped and altered segments of your sample codes provided in class including: TODOTodayII, TouchGestureExperiemnt2, and Snake (the last of which you did not give in class, but I personally asked you for).

**Experiences and Thoughts:** Overall, I had a very fun time coding this project. If I had a bit more time to implement more features, I would like to add more paths for the ball to travel. In the current build, the ball only travels straight down the middle, but I could have made it so that the ball could travel down from the left or right, and maybe I could have it move in a diagonal line down the screen. All of this would require completely rethinking my function of how to tell if the ball hit the right segment. I would also like to try and implement different selectable difficult levels that could increase or decrease the default speed of the ball, the ramp up of the balls speed, or the distance from the top of the screen to the bar.